

### 1. Lungimea unui șir

**unsigned int strlen(char \*sir);**

**Efect:** returnează numărul de caractere al unui șir de caractere, fără a lua în considerare caracterul nul de la sfârșitul șirului

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[256];
  cout<<"dati sirul a=";cin.get(a,256);
  cout<<"sirul are "<<strlen(a)<<" caractere";
}
```

### 2. Operația de copiere

**char \*strcpy(char \*dest,char \*sursa);**

**Efect:** copiază șirul de la adresa sursa la adresa destinație. Copierea se termină la întâlnirea caracterului nul. Funcția returnează adresa șirului destinație. **Simulează operația de atribuire a=b.**

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[100]="crocodil",b[100]="hipopotam";
  strcpy(a,b);
  cout<<"sirul a: "<<a<<endl; //hipopotam
  cout<<"sirul b: "<<b<<endl; //hipopotam
}
```

### 3. Copierea primelor n caractere

**char \*strncpy(char \*dest,char \*sursa,unsigned int n);**

**Efect:** copiază primii n octeți din șirul de la adresa sursă la adresa destinație, fără a adăuga caracterul nul. Funcția returnează adresa șirului destinație. Șirul sursă rămâne nemodificat.

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[100]="crocodil",b[100]="hipopotam";
  strncpy(a,b,4);
  cout<<"sirul a: "<<a<<endl; //hipo
  cout<<"sirul b: "<<b<<endl; //hipopotam
}
```

### 4. Operația de concatenare (adăugare)

**char \*strcat(char \*dest, char \*sursa);**

**Efect:** adaugă șirului de la adresa destinație, înaintea caracterului nul șirul de la adresa sursă. Șirul de la adresa sursă rămâne nemodificat. Operația se numește concatenare. La adresa destinație vom avea șirul destinație urmat de șirul sursă. Șirul destinație are lungimea egală cu suma lungimilor șirurilor.

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[100]="mama ",b[100]="merge";
  strcat(a,b);
  cout<<"șirul a: "<<a<<endl; //mama merge
  cout<<"șirul b: "<<b<<endl; //merge
}
```

### 5. Concatenarea primelor n caractere

**char \*strncat(char \*dest, char \*sursa, unsigned int n);**

**Efect:** adaugă șirului de la adresa destinație, înaintea caracterului nul, primii n octeți ai șirului de la adresa sursă. Șirul de la adresa sursă rămâne nemodificat. Funcția returnează adresa de început a șirului destinație.

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[100]="mama ",b[100]="merge";
  strncat(a,b,3);
  cout<<"șirul a: "<<a<<endl; //mama mer
  cout<<"șirul b: "<<b<<endl; //merge
}
```

### 6. Operația de căutare a unui caracter de la stânga la dreapta

**char \*strchr(char \*sir, int car);**

**Efect:** caută de la stânga la dreapta, caracterul car în șirul de caractere sir. Dacă este găsit, funcția întoarce adresa subșirului care începe cu prima apariție a caracterului citit și se termină cu caracterul nul. Dacă nu este găsit întoarce o expresie de tip char\* cu valoarea 0.

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[100]="crocodil";
  cout<<strchr(a,'o'); //ocodil
}
```

### 7. Operația de căutare a unui caracter de la dreapta spre stânga

**char \*strrchr(char \*sir, int car);**

**Efect:** caută de la dreapta la stânga, caracterul car în șirul de caractere sir. Dacă este găsit, funcția întoarce adresa subșirului care începe cu ultima apariție a caracterului citit și se termină cu caracterul nul. Dacă nu este găsit întoarce o expresie de tip char\* cu valoarea 0.

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[100]="crocodil";
  cout<<strrchr(a,'o'); //odil
}
```

### 8. Identificarea unui subșir

**char \*strstr(char \*sir1,char \*sir2);**

**Efect:** identifică dacă șirul sir2 este subșir(caractere succesive) al șirului sir1. dacă este găsit, funcția returnează adresa sa de început în cadrul șirului s1, altfel returnează 0. Căutarea se face de la stânga la dreapta. Dacă sir2 apare de mai multe ori, returnează adresa primei sale apariții.

**Exemplu:**

```
#include<iostream>
#include<cstring>
void main()
{ char a[100],b[100],*p;
  cin.getlinegets(a,100); cin.getline(b,100);
  p=strstr(a,b);
  if (p) cout<<"este subsir si incepe de la indicele "<<p-a;
  else cout<<"nu este subsir";
}
```

### 9. Analiza sintactică

**char \*strtok(char \*sir1,char \*sir2);**

**Efect:** separă șirul sir1 în entități delimitate de unul sau mai multe caractere din șirul sir2 (acestea având rol de separatori). Apelul funcției se face prima dată sub forma strtok(sir1,sir2) - funcția întoarce adresa primului caracter al primei entități - și a doua oară sub forma strtok(NULL,sir2) și funcția întoarce adresa primului caracter al următoarei entități și după es este adăugat caracterul nul. Când șirul inițial nu mai conține entități, întoarce adresa nulă.

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[100],b[4]=" , ; , *p;
  cin.get(a,100);
  p=strtok(a,b);
  while (p)
  { cout<<p<<endl;
    p=strtok(NULL,b);
  }
}
```

### 10. Compararea a două șiruri

**int strcmp(char \*sir1,char \*sir2);**

**Efect:** compară cele două șiruri de caractere. Valoarea returnată este:

<0 dacă sir1<sir2

=0 dacă sir1=sir2

>0 dacă sir1>sir2

Funcția face distincție între literele mari și literele mici. Compararea șirurilor se realizează comparând de la stânga la dreapta caracter cu caracter. Un șir este mai mic decât altul dacă figurează în dicționar înaintea lui.

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[100],b[100];
  cin.get(a,100); cin.get(b,100);
  int k=strcmp(a,b);
  if (k>0) printf("a>b");
  else if (k==0) printf("a=b");
  else printf("a<b");
}
```

### 11. Compararea fonetică a șirurilor

**int stricmp(char \*sir1,char \*sir2);**

**Efect:** are același efect ca și strcmp dar nu face diferență între literele mari și literele mici.

### 12. Transformarea în litere mari a unui șir

**char \*strupr(char \*s)**

**Efect:** transformă un șir de caractere din litere mici în litere mari. Restul caracterelor rămân nemodificate.

**Exemplu:**

```
#include<iostream>
#include<cstring>
void main()
{ char a[100]="1 crocodil";
  cout<<strupr(a); //1 CROCODIL
}
```

### 13. Transformarea în litere mici a unui șir

**char \*strlwr(char \*s)**

**Efect:** transformă un șir de caractere din litere mari în litere mici. Restul caracterelor rămân nemodificate.

**Exemplu:**

```
#include<iostream>
#include<cstring>
using namespace std;
void main()
{ char a[100]="1 CROCODIL";
  cout<<strlwr(a); //1 crocodil }
```

### Conversie dintr-un șir într-un număr

#### 14. Conversia șir → întreg int

**int atoi(char \*s)**

**Efect:** transformă un șir de caractere într-un întreg (int). Este inclusă în biblioteca <cstdlib>.

**Exemplu:**

```
#include<iostream>
#include<cstdlib>
using namespace std;
void main()
{
    int n;
    char *s="1234.56";
    n=atoi(s);
    cout<<n; // va afisa 1234
}
```

#### 15. Conversia șir → întreg long

**long atol(char \*s)**

**Efect:** transformă un șir de caractere într-un întreg (long). Este inclusă în biblioteca <cstdlib>.

#### 16. Conversia șir → număr real

**double atof(char \*s)**

**Efect:** transformă un șir de caractere într-un număr real. Este inclusă în biblioteca <cstdlib>.

**Exemplu:**

```
#include<iostream>
#include<cstdlib>
using namespace std;
void main()
{
    float n;
    char *s="-4521234.56";
    n=atof(s);
    cout<<n; // va afisa -4521234.56
}
```

### Conversie dintr-un număr într-un șir de caractere

#### 17. Conversie întreg int → șir

**char \*itoa(int val, char \*sir, int baza)**

**Efect:** transformă un număr întreg (int) într-un șir de caractere. Baza reprezintă baza în care este scris noul număr. Este inclusă în biblioteca <cstdlib>.

**Exemplu:**

```
#include<iostream>
#include<cstdlib>
using namespace std;
void main()
{
    int n=12345;
    char s[20];
    itoa(n,s,10);
    cout<<s // va afisa sirul "12345"}
}
```

### 18. Conversie întreg long → șir

**char \*ltoa(long val, char \*sir, int baza)**

**Efect:** transformă un număr întreg (long) într-un șir de caractere.

### 19. Conversie întreg unsigned long → șir

**char \*ultoa(unsigned long val, char \*sir, int baza)**

**Efect:** transformă un număr întreg (unsigned long) într-un șir de caractere.

## Funcții care lucrează cu caractere

Sunt incluse în biblioteca <cctype>. Testează dacă un caracter primit ca parametru îndeplinește o condiție. Returnează 0 dacă acel caracter nu îndeplinește condiția și o valoare diferită de 0 dacă o îndeplinește.

### 20. Testare literă sau cifră

**int isalnum(int c);**

**Efect:** testează dacă un caracter este literă sau cifră

**Exemplu:**

```
#include<iostream>
#include<cctype>
void main()
{ char s='y';
  cout<<isalnum(s); // va afisa o valoare diferita de 0
}
```

### 21. Testare literă

**int isalpha(int c);**

**Efect:** testează dacă un caracter este literă

### 22. Testare cifră

**int isdigit(int c);**

**Efect:** testează dacă un caracter este cifră

**Exemplu:**

```
#include<iostream>
#include<cctype>
void main()
{ char s='y';
  cout<<isdigit(s); // va afisa 0
}
```

### 23. Testare literă mică

**int islower(int c);**

**Efect:** testează dacă un caracter este literă mică

### 24. Testare literă mare

**int isupper(int c);**

**Efect:** testează dacă un caracter este literă mare

### 25. Testare spațiu

**int isspace(int c);**

**Efect:** testează dacă un caracter este spațiu

### 26. Testare cifră în baza 16

**int isxdigit(int c);**

**Efect:** testează dacă un caracter este cifră în baza 16

**Exemplu:**

```
#include<iostream>
#include<cctype>
using namespace std;
void main()
{
    char s='d';
    cout<<isxdigit(s); // va afisa o valoare diferita de 0, deoarece d este o cifra in baza 16
}
```

**27. Transformarea litera mică → litera mare****int toupper(int c);****Efect:** transformă *un caracter* care este litera mică în literă mare**Exemplu:**

```
#include<iostream>
#include<cctype>
void main()
{
    char s='y';
    cout<<toupper(s); // va afisa 'Y'
}
```

**28. Transformarea litera mare → litera mică****int tolower(int c);****Efect:** transformă *un caracter* care este litera mare în literă mică**Exemplu:**

```
#include<iostream>
#include<cctype>
void main()
{
    char s='Y';
    cout<<tolower(s); // va afisa 'y'
}
```